Keyframe Compression and Decompression for Time Series Data based on the Continuous Hidden Markov Model

Tetsunari Inamura[*1] Hiroaki Tanie[*1] Yoshihiko Nakamura[*1*2]

[*1] *Dept. of Mechano-Informatics, The University of Tokyo, JAPAN*
[*2] *CREST program, Japan Science and Technology Corporation, JAPAN*

Abstract

Memory of motion patterns as data, comparison of a new motion pattern with the data, and playback of one from the data are inevitably involved in the information processing of intelligent robot systems. Such computation forms the computational foundation of learning, acquisition, recognition, and generation process of intelligent robotic systems. In this paper, we propose to apply the continuous hidden Markov model to establish the computational foundation, using which one obtains the specified number of keyframes and the their probability distributions. The keyframes are optimally selected to maximize the likelihood. The probability distributions are to be used to compute comparison and playback. The proposed method is applied to the motion data of a humanoid robot as well as the time series image data, and its validity is to be discussed.

1 Introduction

Memory of motion patterns as data, comparison of a new motion pattern with the data, and playback of one from the data are inevitably involved in the information processing of intelligent robot systems. Such computation forms the computational foundation of learning, acquisition, recognition, and generation process of intelligent robotic systems. Motion patterns along with temporal sensory data would be appropriate to describe behaviors of a robot. This is the computational problem of time series data and the subject of the present paper.

The computational problem of time series data would need to consider: (1) efficiency of data compression/decompression, and (2) unification of algorithms for memory (compression), comparison and playback (decompression). The former is mandatory since it determines the volume of database of motion patterns. The latter is not a must, but an important requirement to maintain consistency of the three kinds of computation.

As the regression techniques of time series data, the literature includes the principal component analysis (PCA) [1], and the nonlinear principal component analysis (NPCA) [2]. The both techniques compress the data in the configuration space, but not along the temporal axis.

More recently, different approaches were made for space-time compression from the dynamical point of view. Okada et al.[3] and Ijspeert[4] proposed to memory as a vector field that involves the approximated motion pattern as an attractor. Though less synthetic, a similar approach can be made using an associative memory of recurrent neural network [5]. It is noteworthy that the dynamics based memory can easily compute comparison and playback as well.

A straightforward approach of space-time compression is to determine a limited number of configurations, namely the keyframes, and their temporal positions. The degree of compression can be arbitrary adjusted by setting the number of configurations. The idea was used in the framework of reinforcement learning [6] and simplified the learning computation. The open problems of this straightforward approach are (1) to set a criterion to determine configurations, and (2) to establish a unified computational algorithm of memory, comparison, and playback

In this paper, we propose to apply the continuous hidden Markov model (CHMM) to solve the two open problems simultaneously. The hidden Markov model (HMM) is a doubly structured probabilistic process and has been successfully applied to speech recognition. The field of applications of HMM is growing. However, the role of computation, to the best of the authors' knowledge, is placed at recognition only[7] [8][9]. Using CHMM, one obtains the specified number of keyframes and the their probability distributions. The keyframes are optimally selected to maximize the likelihood by executing the EM algorithm. The probability distributions are to be used to compute comparison and playback. The proposed method is applied to the motion data of a humanoid robot as well as the time series image data, and its validity is to be

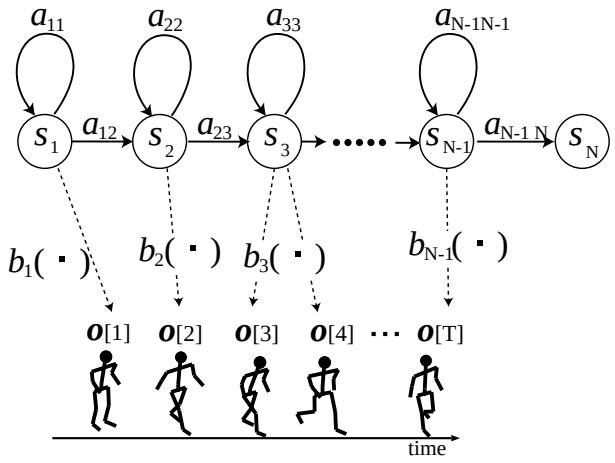

Figure 1: *Continuous Hidden Markov Models and Time-Series Data*

discussed.

2 Keyframe Compression of Time Series Data

2.1 Continuous Hidden Markov Models

The hidden Markov model (HMM) is a probabilistic modeling method and deduces a probabilistic dynamical system that generates the given input data. The HMM consists of a finite number of states $\boldsymbol{S} = \{s_1, \ldots, s_N\}$ and a finite number of connection arcs between each state node. In this framework, state transition occurs probabilistically and delivers a sequence of multi-dimensional vectors as shown in Fig.1.

The probability of the state transition from s_i to s_j is represented by a_{ij}, Matrix $\boldsymbol{A} = \{a_{ij}\}$ is called the state transition probability matrix. Delivery of the multi-dimensional vectors also occurs probabilistically. The process, therefore, is sometimes explained as a doubly probabilistic process.

In the continuous hidden Markov model (CHMM), the domain of output is the continuous vector space, in which any vector possibly becomes an output. The output is determined by the probability density function, which is commonly represented by linear combination of the Gaussian functions as follows:

$$b_i(\boldsymbol{o}) = \sum_{j=1}^{M} c_{ij} \mathcal{N}_{ij}(\boldsymbol{o}; \boldsymbol{\mu}_{ij}, \boldsymbol{\Sigma}_{ij}) \tag{1}$$

where $b_i(\boldsymbol{o})$ is the output probability density function that relates continuous output vector $\boldsymbol{o}$ with the i-th state node s_i. M indicates the number of Gaussian functions used to represent the output probability density function. $\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the Gaussian function:

$$\mathcal{N}(\boldsymbol{o}; \boldsymbol{\Sigma}, \boldsymbol{\mu}) = \frac{\exp\{-\frac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{o}-\boldsymbol{\mu})\}}{\sqrt{(2\pi)^D \det \boldsymbol{\Sigma}}} \tag{2}$$

where $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}$, $\boldsymbol{C} = \{c_{ij}\}$ and D respectively represent the covariance matrix, the mean vector, the mixture coefficient, and the dimension of continuous vector $\boldsymbol{o}$.

The continuous HMM is completely determined by a set of parameter $\{\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{c}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$. $\boldsymbol{\pi}$ is the initial distribution of the starting node. In this paper, $\boldsymbol{\pi}$ has no significance since we adopt the left-to-right model as the structure of state transition network. In the left-to-right model, state transition always starts at the first node, thus the $\boldsymbol{\pi}$ always be $(1, 0, \ldots, 0)$. All the parameters are computed using the Baum-Welch algorithm[10].

2.2 Definition of keyframes

We would like to draw an attention to the meaning of the mean vectors of the Gaussian functions. From the experimental observation we found that pairs of the Gaussian function and its mean vector tend to locate on or near the space-time trajectory of the input data, more interestingly, where they well represent the trajectory as shown in Fig.2. We will discuss this point based on the experimental data in section 4 and 5. This is, however, not surprising and a natural consequence since the Baum-Welch algorithm determines the parameters so that they stochastically best characterize the input data trajectory. This allows us to consider the mean vectors as the keyframes of the trajectory. More details on this matter are in section 2.2.

Therefore, we divide the parameters of continuous HMM, namely, *the dynamical parameter* $\boldsymbol{\lambda}$ and *the keyframe parameter* $\boldsymbol{u}$ as follows.

$$\boldsymbol{\lambda} \stackrel{\text{def}}{=} \{\boldsymbol{A}, \boldsymbol{c}\} \tag{3}$$

$$\boldsymbol{u} \stackrel{\text{def}}{=} \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\} \tag{4}$$

The number of ketframes obviously coincides with that of the Gaussian distributions, and it is reserved for us as a free design parameter.

Computation of memory (compression) for learning and aquisition Memory computation is to compute the parameters of CHMM ($\boldsymbol{\lambda} = \{\boldsymbol{A}, \boldsymbol{B}\}$). It implies the data compression of time series data as we have already viewed. A simple learning algorithm could be to recover a single set of parameters from the database of many trials of the same motion pattern, for example. One could develop a much sophisticated

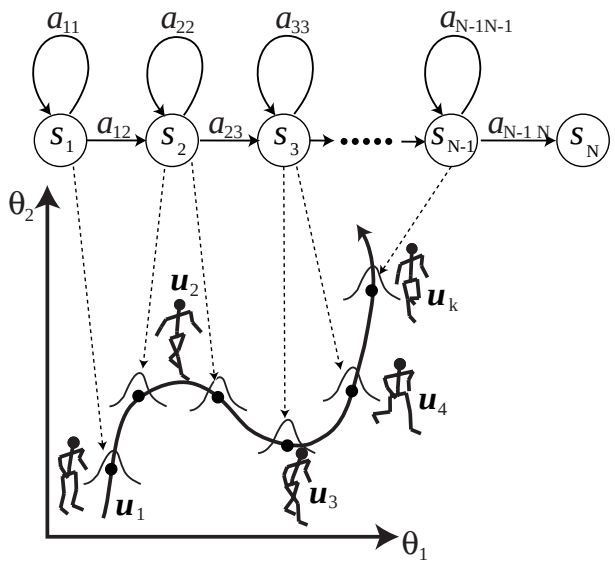

Figure 2: *Gaussian distribution and state space*

learning or acquisition algorithm based on the memory computation. The practice of memory computation can be done by one of the expectation-maximizing (EM) algorithms [10], which are well studied and established for computational efficiency.

Computation of playback (decompression) for generation Computation of playback is to generate a time series of data from the HMM. This is a complimentary computation of memory computation and, therefore, corresponds to decomposition. This will be in detail discussed in the section that follows.

Computation of comparison for recognition When a new time series of data

$$\boldsymbol{O} = \left[\begin{array}{cccc} \boldsymbol{o}[1] & \boldsymbol{o}[2] & \ldots & \boldsymbol{o}[T] \end{array} \right] \tag{5}$$

is given, the HMM can compute likelihood (or probability) $p(\boldsymbol{O}|\boldsymbol{\lambda}, \boldsymbol{u})$ in which the HMM with $\boldsymbol{\lambda}, \boldsymbol{u}$ actually generates $\boldsymbol{O}$. This computation is done by a well-known forward algorithm [10]. The likelihood is larger if the new data is close to the original data used for memory computation. It is smaller if it is far from the original data. Accordingly, one can compare a time series of data with the other data. Such comparison straightforwardly offers the means of recognition of similar time series data.

3 Averaging Method for Computation of Playback (Decomposition)

In this section, we develop an algorithm for the computation of playback in addition to the other two computations explained in the previous section. Although the HMM intrinsically holds the original time series data within the parameters, reproducing it from the parameter is not trivial.

It is direct and involves less computation to follow the double stochastic processes to generate a time series of data from the given HMM.

Here, there are sway among each reproduction processes because of the property of stochastic models. The time length of the motions T_i always changes by the state transition probability $\boldsymbol{A}$, the value for each moment of the time-series data also always changes by the output probability $b_i(\boldsymbol{o})$. We propose an average strategy in order to cancel these sway.

***step*1** Getting a state transition sequence $\boldsymbol{Q} = \left[s_{k[1]}, s_{k[2]}, \cdots, s_{k[T]}\right]$, $(k[i] \in \{1, 2, \ldots, N\})$ with a trial of the state transition process.

***step*2** An average of transition sequence $\hat{\boldsymbol{Q}}$ is calculated by n_q times repetition $(\boldsymbol{Q}_1, \cdots, \boldsymbol{Q}_{n_q})$ of the ***step*1**.

***step*3** Output time-series data $\boldsymbol{O}$ is calculated by a trial of output according to the average transition sequence $\hat{\boldsymbol{Q}}$.

***step*4** $\boldsymbol{O}_1, \cdots, \boldsymbol{O}_{n_o}$ is calculated by n_o times repetition from ***step*1** to ***step*3**.

***step*5** Average time-series data $\hat{\boldsymbol{O}}$ is calculated by the $\boldsymbol{O}_1, \cdots, \boldsymbol{O}_{n_o}$ after regularization of the time length.

where n_q, n_o are decided experimentally.

In the ***step*2**, the average process treats discrete state transition as continuous time-series function, which we call as continuous state transition. The function q is designed using interpolation of following set of discrete values.

$$q_i(\Delta t \cdot j) = k[j] \tag{6}$$

$\hat{q}$ which is the average of q is calculated after the calculation of time canonicalization q'.

$$q_i'(\tau) = q_i(T_i \cdot \tau) \tag{7}$$

$$\hat{q}(\tau) = \sum_i^{n_q} q_i(\tau) \cdot \frac{1}{n} \tag{8}$$

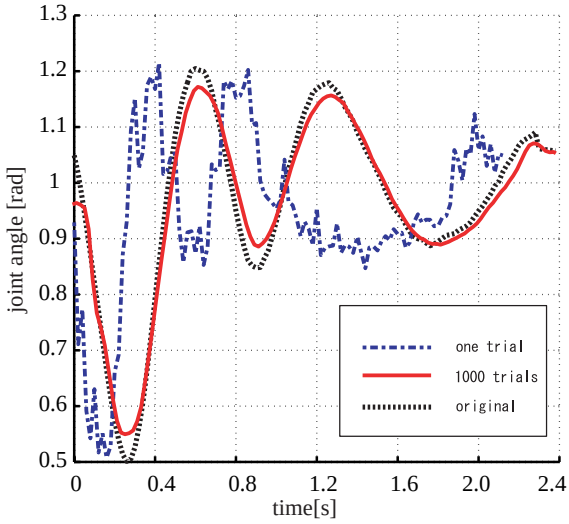

Figure 3: *Original time-series data and generated data*

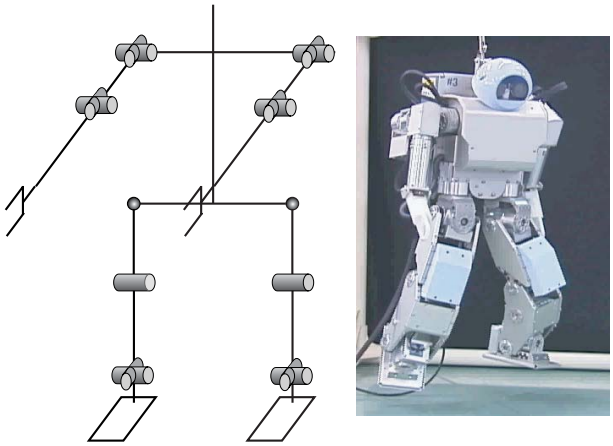

Figure 4: *A humanoid used in the application*

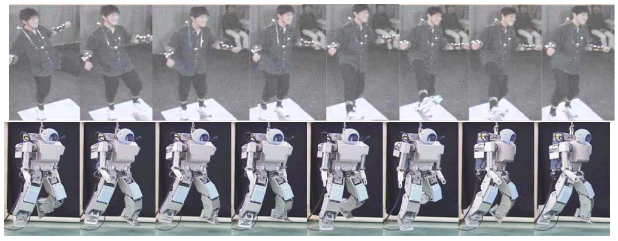

Figure 6: *Observed humans' stepping performance (upper), and a result of step motion generation on a real humanoid robot (lower)*

where Δt is time step length which is corresponds to each state transition, τ is a parameter for time canonicalization ($0 \leq \tau \leq 1$), j is the discrete time index. Finally, $\hat{Q}$ is calculated by integer approximation and time length decomposition as follows

$$\hat{T} = \sum_{i}^{n_q} T_i \cdot \frac{1}{n}. \tag{9}$$

Figure 3 shows the example decomposition result using the method. Target data is joint angle data of a humanoid robot. The dot-line indicates the original time-series data, the dashed line indicates a result of single output trial (Q_i). There are deviance for time direction and value direction, however, the deviance for time direction was canceled using ***Step2***, and the deviance for value direction was also canceled using ***step5***. Final output result is shown as solid line. As the figure shows, output time-series data is similar to the original time-series data.

4 Application for Whole-body Motion of Humanoids

To describe the whole body motion, we have adopted simple joint angles for each moment as a keyframe, that is, the keyframe elements are represented as follows:

$$\boldsymbol{u} \stackrel{\text{def}}{=} (\theta_1, \theta_2, \ldots, \theta_D)^T, \tag{10}$$

where D is the number of joints which corresponds to the D in Eq.2. A humanoid robot shown in Fig.4 is used for the application. The humanoid has 20 joints, thus the number of dimension of the keyframe D is 20.

We have confirmed the performance of our method for imitation learning framework which have been proposed by us [11]. Four kinds of motions; walking, squat, picking up, and Cossack dancing, were recorded by a motion capturing system. Using the system, joint angle data for 20 DOFs are directly observed. The time period of each motion is about 2[sec] with sampling time 20[msec].

After the 50 observations, motion generation process is executed. Figure 5 shows the acquired keyframes. Each dot mark indicate the keyframes, solid line indicate the original motion's trajectory. It is difficult to represent the whole dimension, three joints; hip(pitch axis), knee and ankle(pitch axis) are drawn.

As the figure shows, the keyframes are located near the original motion. Additionally, keyframes are apt to be distributed to important and impressive state point like a turning point of the motion. These properties show the advantage of our method.

Using the method, we also performed the imitation

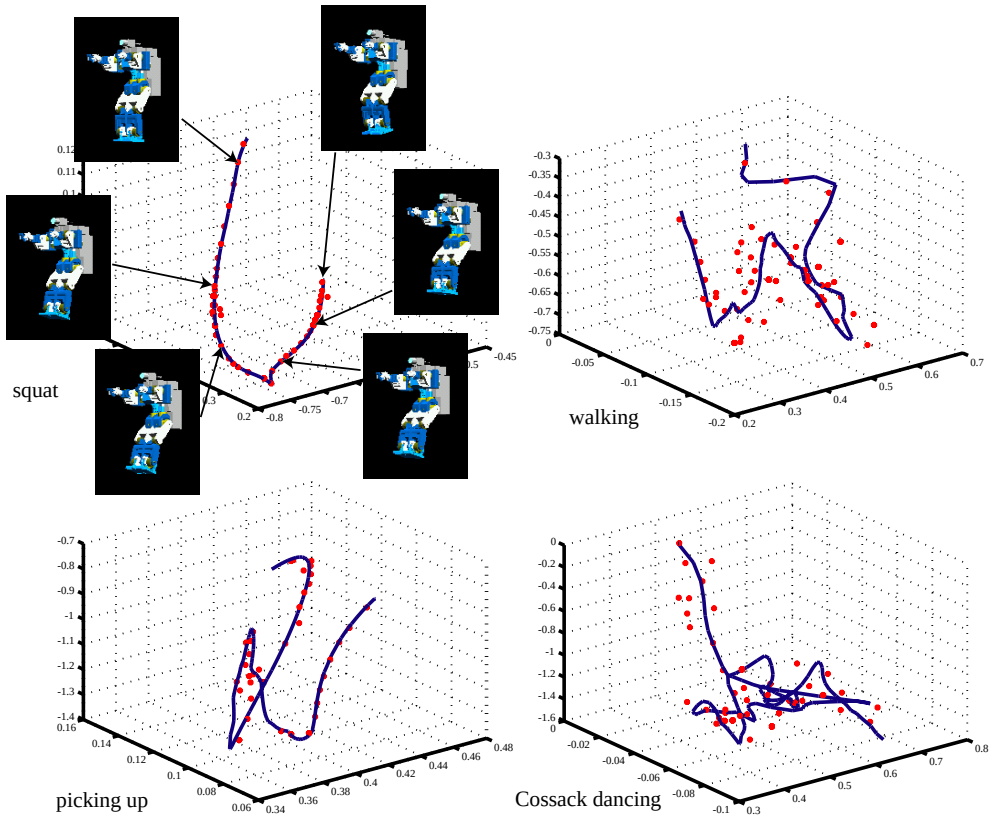

Figure 5: *A result of motion elements acquisition against four kinds of motion*

learning experiment against humans' stepping motions as shown in Fig.6. Lower of the figure shows experimental result on a humanoid robot.

5 Application for time-series image

Next, we have applied the proposed framework to time-series image processing. A target task is recognition of time-series image data and navigation on mobile robots.

As the size of image for each sampling time is $N \times M$ [pixels], the size of output vector $\boldsymbol{o}_i$ for continuous HMMs becomes $N \times M$. Under the condition, the *impressive elements* becomes as follows

$$\boldsymbol{u}_i = \begin{pmatrix} I_{11} \\ I_{12} \\ \cdots \\ I_{NM} \end{pmatrix} \tag{11}$$

where I_{ij} indicates the brightness on (i, j) pixel.

As an experiment, image sequence shown in Fig.7 have been captured during running in corridor environment by manual operation. The size of the image is $N = 32, M = 32$, then D the size of image vector for each moment is 1024. A continuous HMM which has five nodes and four mixture components for each node, was learnt. The time cost for the learning was about 3 [min].

Acquired view keyframes are shown in Fig.8. Impressive scene such as when the robot pass through by the door, when the robot observed a light, when the inside of the room started to be seen, are acquired by the continuous HMM. There scene image can be applied to not only recognition process but also navigation task and saving memory using by a method proposed in [12].

One of the weakness of movie playing using MPEG is that integration from beginning image is needed when the user try to play from any frame. Owing to the proposed method, effective playing might be executed using keyframe representation and dynamics decomposition using HMMs.

6 Conclusions

In this paper, we proposed that the Hidden Markov Models which usually used as a recognition tool, can be applied to compression of time-series data. In this

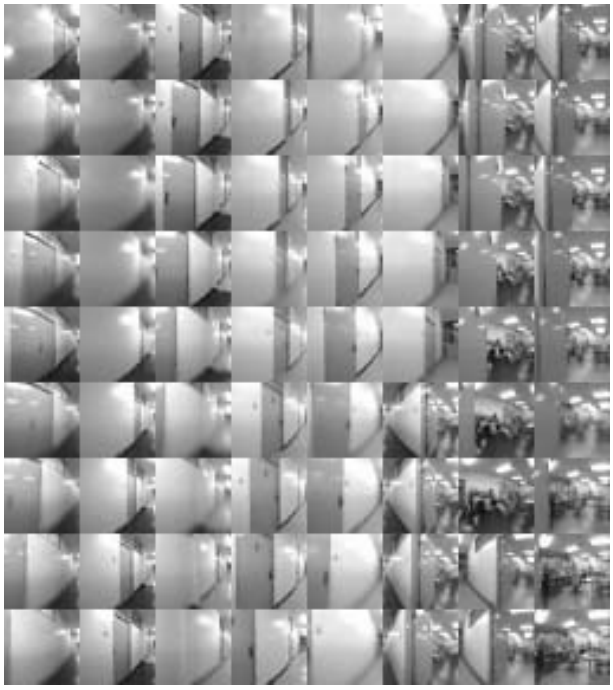

Figure 7: *Observed image data on a mobile robot*

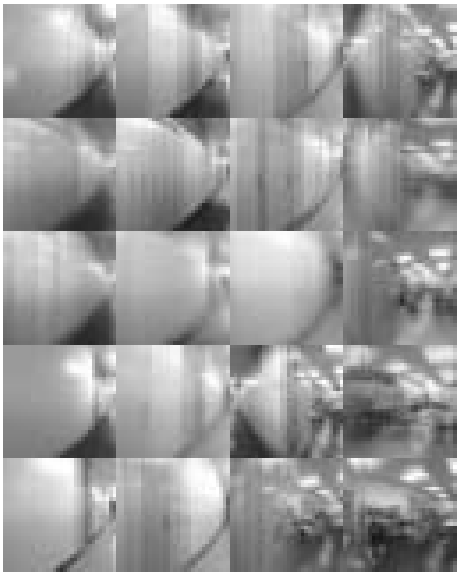

Figure 8: *Acquired image elements*

method, not only data compression but also choosing impressive moment and design the keyframe for the recognition is performed, because the advantage of dynamics approach that dynamics information is contained in the compression representation, and another advantage of keyframe approach that the impressive moments are automatically selected for the recognition and generation of time-series data.

Moreover, not only time-series data recognition, but also motion generation on humanoid robots and CG character, and navigation on mobile robots are available with help of the method through experiments. Especially, effective result have been acquired that the method can show good performance against complex tasks such as navigation task which ought to treats large size of dimension data. We think the applicable area would spread over various time-series data processing on intelligent robots.

Acknowledgement

This research was supported by the Core Research for Evolutional Science and Technology (CREST) program of the Japan Science and Technology Corporation (PI: Y. Nakamura).

References

[1] B. C. Moore. Principal component analysis in linear systems: Controllability, observability and model reduction. *IEEE Transaction*, AC-37(1):17–32, 1981.

[2] Mark Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37:233–243, 1991.

[3] Masafumi Okada, Koji Tatani, and Yoshihiko Nakamura. Polynomial design of the nonlinear dynamics for the brain-like information processing of whole body motion. In *Proc. of IEEE Int'l Conf. on Robotics and Automation*, pages 1410–1415, 2002.

[4] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of IEEE Int'l Conf. on Robotics & Automation*, pages 1398–1403, 2002.

[5] Satoshi Murakami and Masahiko Morita. Top-down and bottom-up processing of spatiotemporal patterns in a fully recurrent network of nonmonotonic neurons. In *Proceedings of the 1999 Int'l Conf. on Neural Information Processing*, volume 3, pages 1118–1122, 1999.

[6] Jun Morimoto and Kenji Doya. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Robotics and Autonomous Systems*, 36:37–51, 2001.

[7] P.K. Pook and D.H. Ballard. Recognizing teleoperated manipulations. In *IEEE Int'l Conf. on Robotics and Automation*, pages 578–585, 1993.

[8] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, pages 379–385, 1992.

[9] Koichi Ogawara, Jun Takamatsu, Hiroshi Kimura, and Katsushi Ikeuchi. Modeling manipulation interactions by hidden markov models. In *Proc. of 2002 IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 1096–1101, 2002.

[10] Steve Young et al. *The HTK Book*. Microsoft Corporation, 2000.

[11] Tetsunari Inamura, Iwaki Toshima, and Yoshihiko Nakamura. Acquisition and embodiment of motion elements in closed mimesis loop. In *the Proc. of IEEE Int'l Conf. on Robotics & Automation*, pages 1539–1544, 2002.

[12] Yoshio Matsumoto, Masayuki Inaba, and Hirochika Inoue. View-based approach to robot navigation. In *Proceedings of 2000 IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 1702–1708, 2000.